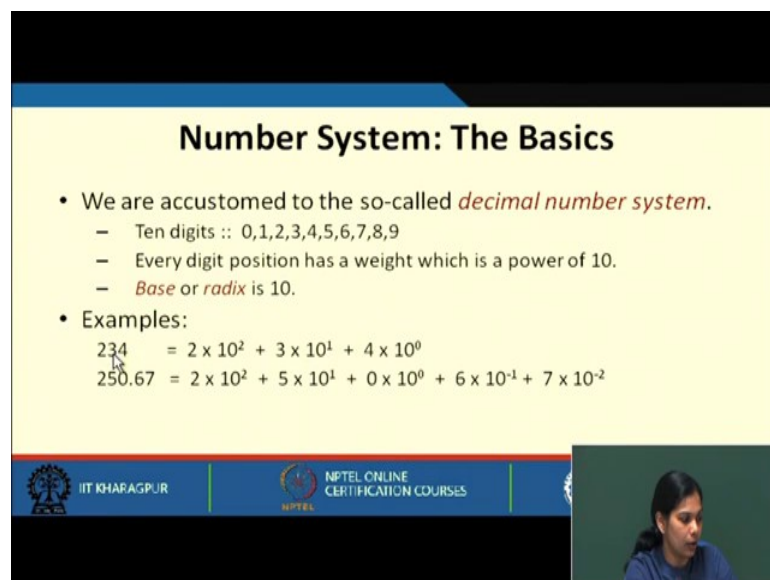


Computer Architecture and Organization
Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture - 06
Number Representation

Welcome back. We shall now be moving on with the detailed discussion on various aspects of computer architecture and organization. As we know that typically digital circuits work on binary number system which can ultimately be mapped to some tiny electronic switches. So, in this lecture we will be considering this number representation in computer systems.

(Refer Slide Time: 00:54)



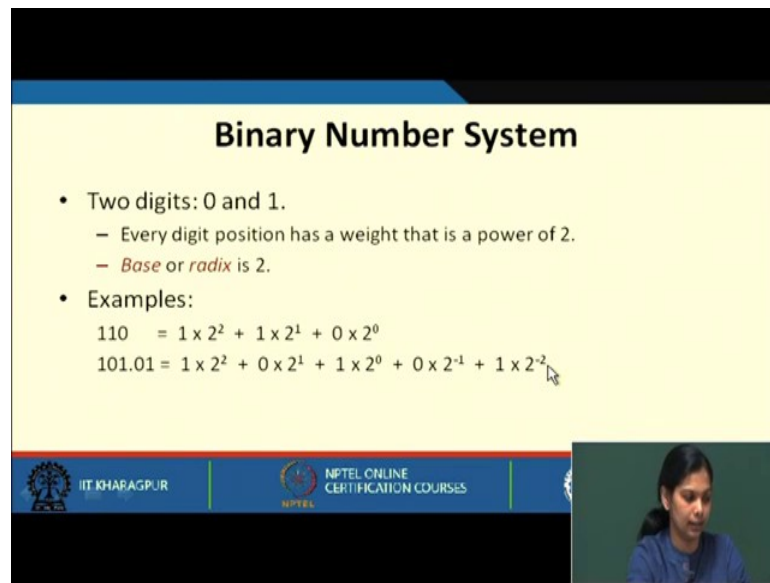
Number System: The Basics

- We are accustomed to the so-called *decimal number system*.
 - Ten digits :: 0,1,2,3,4,5,6,7,8,9
 - Every digit position has a weight which is a power of 10.
 - *Base or radix* is 10.
- Examples:
 $234 = 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$
 $250.67 = 2 \times 10^2 + 5 \times 10^1 + 0 \times 10^0 + 6 \times 10^{-1} + 7 \times 10^{-2}$

The slide footer includes the logos of IIT Kharagpur and NPTEL Online Certification Courses, along with a small video inset of the professor.

We all are accustomed with the so called decimal number system. In decimal number system, we represent ten digits 0 to 9; and every digit position has a weight which is a power of 10, called base or radix. Let us take an example 234, so, it can be represented the weighted representation like this: 4×10^0 in the unit position, then 3×10^1 in the tenth position, and 2×10^2 in the hundredth position. Similarly, we can also represent both integer part and fractional part, where for the integer part we multiplied this number with 10^0 , 10^1 , and 10^2 and here we start with -1. So, we multiply 6×10^{-1} , 7×10^{-2} , and so on.

(Refer Slide Time: 02:26)



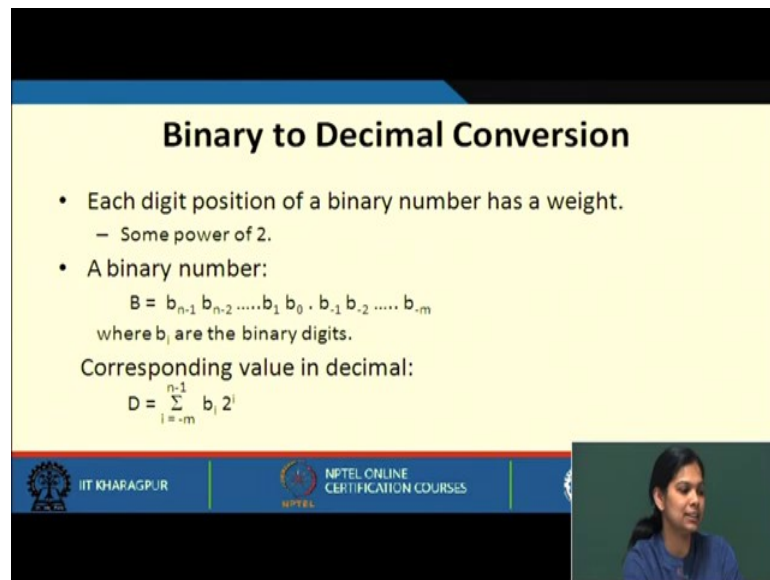
Binary Number System

- Two digits: 0 and 1.
 - Every digit position has a weight that is a power of 2.
 - *Base or radix* is 2.
- Examples:
 - $110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 - $101.01 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a woman in the bottom right corner.

Coming to binary number system, we all know we have two digits 0 and 1. And every digit position has a weight that is power of 2; in decimal number system we have power of 10; and here we have power of 2. So, the base or radix here is 2. Let us see how we can represent a binary number. So, 1 1 0 is this number: 0 will be multiplied with 2^0 , the next number will be multiplied with 2^1 , and the next will be multiplied with 2^2 . So, this is the weighted representation of this number. Similarly, we can also represent a fractional part. So, 101.01, it can be represented, so this will be 2^0 , this will be 2^1 , and this will be 2^2 in the same way. And again in the fractional part we will be multiplying with 2^{-1} , 2^{-2} , and so on. So, this will be $0 \times 2^{-1} + 1 \times 2^{-2}$ and so on.

(Refer Slide Time: 04:13)



Binary to Decimal Conversion

- Each digit position of a binary number has a weight.
 - Some power of 2.
- A binary number:
 $B = b_{n-1} b_{n-2} \dots b_1 b_0 . b_{-1} b_{-2} \dots b_{-m}$
where b_i are the binary digits.

Corresponding value in decimal:
$$D = \sum_{i=-m}^{n-1} b_i 2^i$$

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a woman in the bottom right corner.

Now, let us see some conversions, like binary to decimal, decimal to binary, binary to hexadecimal, hexadecimal to binary. So, we will be looking into all those things. So, when we are doing binary to decimal conversion, each digit position of a binary number has a weight and that weight is some power of 2. So, our binary number can be represented like this, this is the integer part and this is the fractional part. The integer part starts from b_0, b_1, b_{n-1} ; and the fractional part b_{-1}, b_{-2} to b_{-m} , where b_i are the binary digits.

So, the corresponding value in decimal will be all these binary digits multiplied by 2^i . So, if this digit is b_0 , then $b_0 \times 2^0$. If it is b_1 , then it will be $b_1 \times 2^1$ and so on, and here $b_{-1} \times 2^{-1}$, and so on, till b_{-m} . So, the corresponding value in decimal can be represented by this summation.

(Refer Slide Time: 06:14)

Some Examples

- $101011 \rightarrow 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 43$
 $(101011)_2 = (43)_{10}$
- $.0101 \rightarrow 0x2^{-1} + 1x2^{-2} + 0x2^{-3} + 1x2^{-4} = .3125$
 $(.0101)_2 = (.3125)_{10}$
- $101.11 \rightarrow 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2} = 5.75$
 $(101.11)_2 = (5.75)_{10}$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us take some examples, which will make this clear. So, we have a number 1 0 1 0 1 1, this is a binary number, how we can convert it into a decimal number? We start with multiplying 1 with 2^0 and so on, and we get 43 in decimal. Similarly if we have a fractional part to convert it we have to multiply this with 2^{-1} , and so on. Similarly, if we have a both integer part and fractional part in the same way we can consider the first part as like this and the next part like this and finally, we get a value like this.

(Refer Slide Time: 07:43)

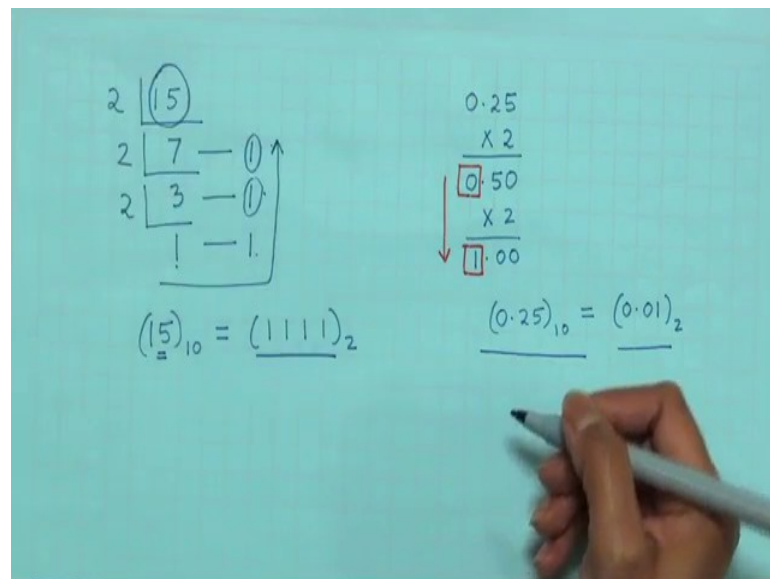
Decimal to Binary Conversion

- Consider the integer and fractional parts separately.
- For the integer part:
 - Repeatedly divide the given number by 2, and go on accumulating the remainders, until the number becomes zero.
 - Arrange the remainders *in reverse order*.
- For the fractional part:
 - Repeatedly multiply the given fraction by 2.
 - Accumulate the integer part (0 or 1).
 - If the integer part is 1, chop it off.
 - Arrange the integer parts *in the order* they are obtained.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, let us consider decimal to binary conversion. So, we have seen binary to decimal conversion. Now, we will be seeing decimal to binary conversion. Consider the integer and fractional part separately. Here we have to consider the integer and fractional part separately let us see how. So, what we do for the integer part, for the integer part we repeatedly divide the given number by 2 and we go on accumulating the remainders until the number becomes 0.

(Refer Slide Time: 08:20)



So, let us take an example. So, here we can see that the number is 15. What we say that we repeatedly divide the given number by 2, so we are dividing the given number by 2 we are getting a quotient and we are getting a remainder. And we are going on accumulating the remainder that is here it is 1; again we do it, we get here 3 and the remainder becomes 1. Again we divide it and we get the remainder as 1 and the quotient as 1 as well. And what we do finally, we arrange it in reverse order. So, in reverse order when we arrange this 15 in decimal is now converted to 1 1 1 1 in binary. So, here this is what it has been said that for the integer part we repeatedly divide a given number by 2 and we accumulate the remainder until the number becomes 0 and then we arrange it in reverse order the way.




Next, for the fractional part, we repeatedly multiply the given fraction by 2, and then we accumulate the integer part. If the integer part is 1, we chop it off. And then finally, while arranging we will be arranging the order the integer parts in the same order.

So, now let us see this with an example of 0.25. We multiply it by 2 and we get 0.50. So, here 0 is accumulated this is the integer part which is 0. Next, again we take the fractional part we do not do anything with this part. So, we take the fractional part, and again we multiply it by 2. And what we get we get 1.00, where we get this fractional, this integer part as 1. So, here we can say that 0.25 in decimal can be represented as 0.01. So, this point remains and what we got is 01 we are not writing this in reverse order rather we are writing it as 0 1 0 1. So, this is the decimal this is the binary representation of this particular decimal fractional part. This is how we do it. And we can continue doing it because it is not necessary that we will get 0 0, we can get any number. So, we can do it for some number of times and we get the result.

(Refer Slide Time: 11:49)

Examples

$\begin{array}{r} 2 \overline{) 239} \\ \underline{2 \ 119} \quad \dots 1 \\ 2 \ 59 \quad \dots 1 \\ \underline{2 \ 29} \quad \dots 1 \\ 2 \ 14 \quad \dots 1 \\ \underline{2 \ 7} \quad \dots 0 \\ 2 \ 3 \quad \dots 1 \\ \underline{2 \ 1} \quad \dots 1 \\ 2 \ 0 \quad \dots 1 \end{array}$ <p>$(239)_{10} = (11101111)_2$</p>	$\begin{array}{r} 2 \overline{) 64} \\ \underline{2 \ 32} \quad \dots 0 \\ 2 \ 16 \quad \dots 0 \\ \underline{2 \ 8} \quad \dots 0 \\ 2 \ 4 \quad \dots 0 \\ \underline{2 \ 2} \quad \dots 0 \\ 2 \ 1 \quad \dots 0 \\ \underline{2 \ 0} \quad \dots 1 \end{array}$ <p>$(64)_{10} = (1000000)_2$</p>	$\begin{array}{l} .634 \times 2 = 1.268 \\ .268 \times 2 = 0.536 \\ .536 \times 2 = 1.072 \\ .072 \times 2 = 0.144 \\ .144 \times 2 = 0.288 \\ \vdots \\ (.634)_{10} = (.10100\dots)_2 \end{array}$	$\begin{array}{l} 37.0625 \\ (37)_{10} = (100101)_2 \\ (.0625)_{10} = (.0001)_2 \\ \therefore (37.0625)_{10} = \\ (100101.0001)_2 \end{array}$
--	---	---	--

Now, let us see some more examples. Here in the same way 239 in decimal can be converted to binary where we get 1 1 1 0 1 1 1 1, which we write in reverse order. Similar way, let us take another example which is 64 we go on dividing with the number by 2 and we accumulate the remainder and finally, what we get is 1 0 0 0 0 0 0.

Let us take some example with the fractional part. So, here we see that 0.634 in decimal how we can convert it into binary. In the same way we multiply it by 2 and we keep the integer part. And then we take the fractional part again and keep on multiplying with 2. And again after multiplying in we keep the integer part and again with this fractional part we multiply it again with 2, and keep on doing it.

Let us say we have done it till 1 2 3 4 and 5 times and we write it in this order. So, the value will be point 1 0 1 0 0 and if you want to do it for some more you can also do that. So, this is how we can convert a fractional decimal part to binary. Similarly, we can take an example of a number, which is having both integer part and fractional part. So, the integer part can be converted and it is represented by this and the fractional part is converted and it is represented by this, and finally you can write the answer in this fashion.

(Refer Slide Time: 14:19)

Hexadecimal Number System

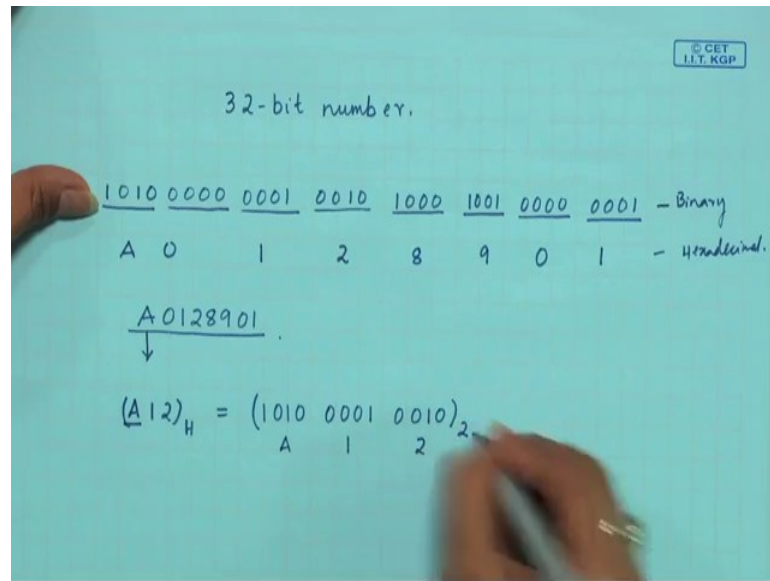
- A compact way to represent binary numbers.
 - Group of four binary digits are represented by a hexadecimal digit.
 - Hexadecimal digits are 0 to 9, A to F.

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, coming to hexadecimal number system. Now, why do we require hexadecimal number system? Now, let us say you have to represent a 32-bit number or a 64-bit number. So, if you have to represent a 64-bit number in binary, you have to represent it as 0 1 0 1 up to 64 bits, or if it is a 32-bit number then you have to write that 0 1 0 1 whatever is the number till 32 bits. So, it will be really very long. Hexadecimal number system is a compact way to represent binary numbers. This is an efficient way where what we do basically is group four binary digits and represent by a hexadecimal digit. And the hexadecimal digits are 0 to 9, and A to F. 0 to 9 we already know and starting from 10 is represented as A, 11 is represented as B, 12 as C, 13 as D, 14 as E, and 15 as F. So, we have these representations starting from 0 to F in hexadecimal number. So, a group of four numbers will be replaced with a single digit. If we encounter 1 0 1 0, we will replace this with the digit A.

(Refer Slide Time: 16:28)



So, if we have to represent a 32-bit number, we can represent by only 8 digits. How, because 1010 is A, 0000 is 0, 0001 is 1, 0010 is 2, this is 8, this is 9, this is 0, this is 1. So, this entire number A0128901 is the hexadecimal representation of this 32-bit number.

(Refer Slide Time: 18:14)

Binary to Hexadecimal Conversion

- For the integer part:
 - Scan the binary number from *right to left*.
 - Translate each group of four bits into the corresponding hexadecimal digit.
 - Add *leading* zeros if necessary.
- For the fractional part:
 - Scan the binary number from *left to right*.
 - Translate each group of four bits into the corresponding hexadecimal digit.
 - Add *trailing* zeros if necessary.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, we will move on. So, let us see how we can convert this binary number to hexadecimal. So, for the integer part, we scan the binary number from right to left, and we translate each group of four bits into the corresponding hexadecimal digit. And if

required we add leading zeros that means, if it is required that we need to add some more bits to make it a group of four then we can add those bits.

So, for the fractional part, what we do we scan the binary number from left to right. So, there is a difference between the integer part and your fractional part. In the integer part, we are doing from right to left; and in the fractional part, we are doing from left to right. And what we do we translate each group of four bits into the corresponding hexadecimal digit that I have already shown you. And for doing so if it is required then what we do we add trailing zeroes into it.

(Refer Slide Time: 19:54)

Examples

1. $(\underline{1011} \underline{0100} \underline{0011})_2 = (B43)_{16}$
2. $(\underline{010} \underline{1010} \underline{0001})_2 = (2A1)_{16}$ *Two leading 0s are added*
3. $(\underline{.1000} \underline{010})_2 = (.84)_{16}$ *A trailing 0 is added*
4. $(\underline{101} . \underline{0101} \underline{111})_2 = (5.5E)_{16}$ *A leading 0 and trailing 0 are added*

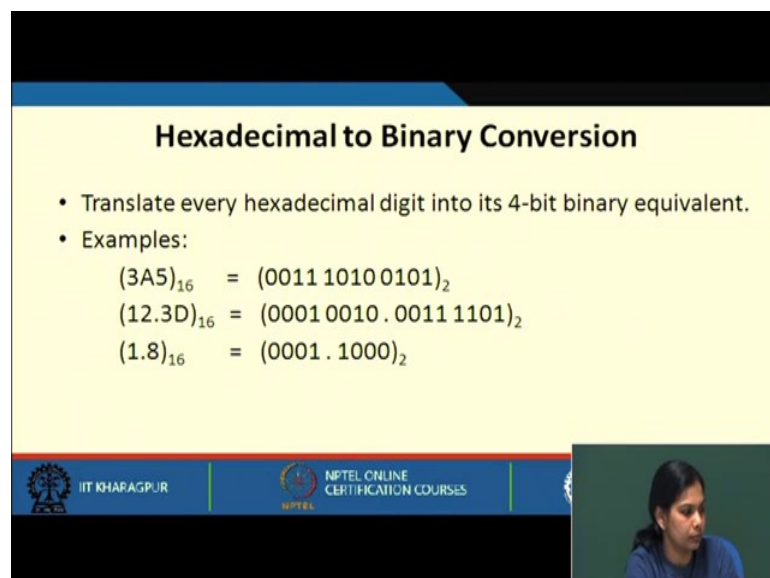
Let us take some more examples. Let us say we have a number which is represented in binary 1010. So, we group it from we group this from right to left. So, first we group this four bit, next four bit, then next four bit and then we find out that we have a group of three groups having four-four bits. So, what is 1011 -- it is B. What is 0100 -- it is 4 in hexadecimal; and 0011 is 3. So, we have converted a binary number into hexadecimal representation.

Similarly, here you see you do not have a group of four. So, what you have to do you have to add some zeros in the beginning. So, here we group it four here also we group it four, but here we cannot group it four. So, you add 2 more zeros. So, 2 leading zeros are added to make it 0010 which is equivalent to 1010 is A, and 0001 is 1. Similarly, for the fractional part we scan from left to right. So, scanning this from left to right, so we have

to group into four-four bits. So, this is the first four bits and this is the next four bits. To make it four, we have to add a trailing zero. So, a trailing zero is added to make it a group of four bits. So, this 1000 is 8 and 0100 is 4 which in hexadecimal.

Similarly, for this is the integer part and this is the fractional part. For the integer part, we have to add a leading 0 which makes it 0101, 0101, so which is 5 and then for this we have to add a trailing 0. So, if we add a trailing 0 this will make it as 5, and this is E. So, this is how we convert a binary number into hexadecimal number for simply representing it in a compact fashion.

(Refer Slide Time: 22:22)



Hexadecimal to Binary Conversion

- Translate every hexadecimal digit into its 4-bit binary equivalent.
- Examples:
 - $(3A5)_{16} = (0011\ 1010\ 0101)_2$
 - $(12.3D)_{16} = (0001\ 0010 . 0011\ 1101)_2$
 - $(1.8)_{16} = (0001 . 1000)_2$

The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of a woman in the bottom right corner.

Now, in a similar way, we can also perform hexadecimal to binary conversion. So, we can convert a number from hexadecimal to binary, by translating every hexadecimal digit into 4 binary digit.

(Refer Slide Time: 24:21)

How are Hexadecimal Numbers Written?

- Using the suffix "H" or using the prefix "0x".
- Examples:
 - ADDI R1,2AH // Add the hex number 2A to register R1
 - 0x2AB4 // The 16-bit number 0010 1010 1011 0100
 - 0xFFFFFFFF // The 32-bit number for the all-1 string

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, how are hexadecimal numbers written? So, like we have to represent a number as hexadecimal otherwise it will be difficult for the computer to understand. So, to do that you can either add a suffix H or you can use the prefix 0x. So, this here we are representing this 2A is a hexadecimal number, where H is added as a suffix. And here 2AB4 is a hexadecimal number --- this is a 16-bit hexadecimal number and it is added with a prefix 0x. And this is a 32-bit hexadecimal number with all 1s, so it is represented as FFFFFFFF.

(Refer Slide Time: 25:33)

Unsigned Binary Numbers

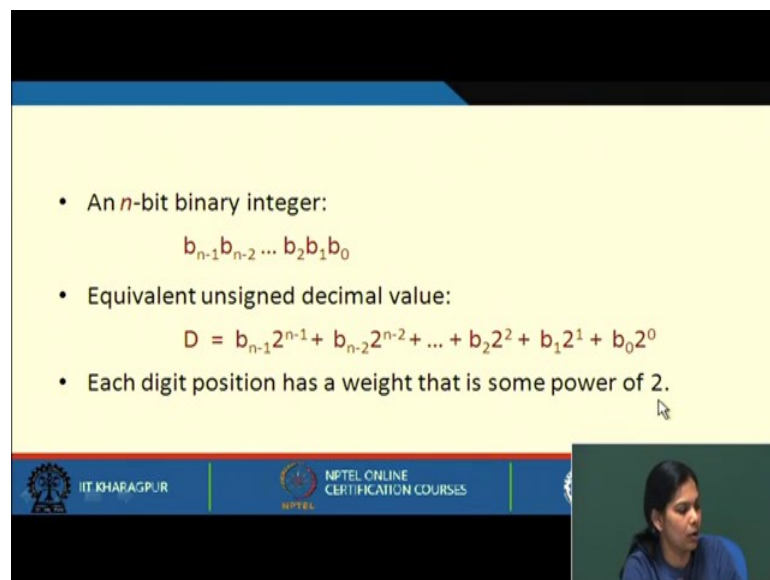
- An n -bit binary number can have 2^n distinct combinations.
 - For example, for $n=3$, the 8 distinct combinations are: 000, 001, 010, 011, 100, 101, 110, 111 (0 to $2^3-1 = 7$ in decimal).

Number of bits (n)	Range of Numbers
8	0 to 2^8-1 (255)
16	0 to $2^{16}-1$ (65535)
32	0 to $2^{32}-1$ (4294967295)
64	0 to $2^{64}-1$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, an n -bit binary number can have 2^n distinct combination. For example, if you consider $n = 3$, so we will have 2^3 combination; starting from 0 to 2^n-1 . So, the first is 000, next is 001, 010, 011 and so on till 7. So, similarly for an 8-bit number, it will be 0 to $2^8-1 = 255$; for 16-bit it is 0 to $2^{16}-1$ and so on.

(Refer Slide Time: 26:50)



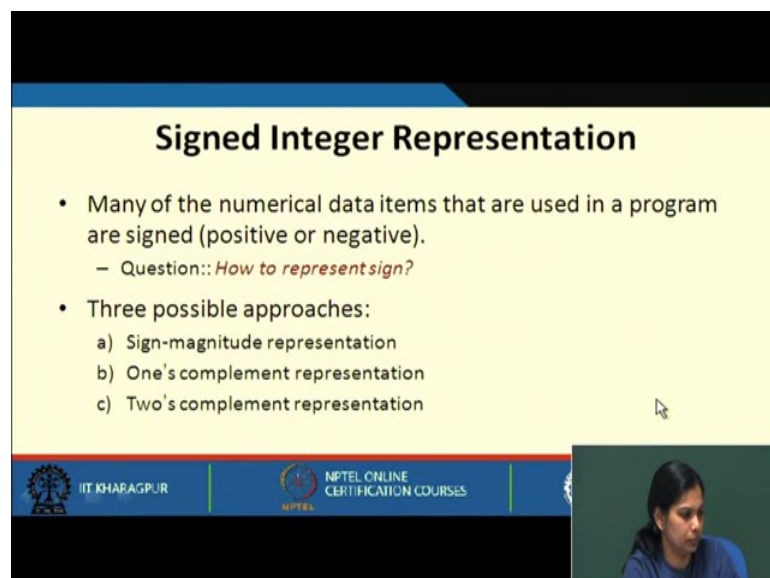
The slide contains the following text:

- An n -bit binary integer:
$$b_{n-1}b_{n-2} \dots b_2b_1b_0$$
- Equivalent unsigned decimal value:
$$D = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_22^2 + b_12^1 + b_02^0$$
- Each digit position has a weight that is some power of 2.

The slide footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a woman speaking.

So, so n bit binary integers can be represented like this, and the equivalent unsigned decimal number is represented by this; it is multiplied by 2 to the power 0. So, each digit position has a weight that is some power of 2.

(Refer Slide Time: 27:09)



The slide is titled "Signed Integer Representation" and contains the following text:

- Many of the numerical data items that are used in a program are signed (positive or negative).
– Question:: *How to represent sign?*
- Three possible approaches:
 - a) Sign-magnitude representation
 - b) One's complement representation
 - c) Two's complement representation

The slide footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a woman speaking.

Now, let us come to signed integer representation. So, in signed integer representation, how do we represent negative numbers, how do we represent both positive and negative numbers? So, here the question arises how to represent the sign bit. There are three approaches that are followed; one is sign magnitude representation another is one's complement representation and another is two's complement representation.

(Refer Slide Time: 27:42)

(a) Sign-magnitude Representation

- For an n-bit number representation:
 - The most significant bit (MSB) indicates sign (0: positive, 1: negative).
 - The remaining (n-1) bits represent the magnitude of the number.
- Range of numbers: $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

Diagram illustrating the bit layout for sign-magnitude representation:

b_{n-1}	b_{n-2}					b_1	b_0
-----------	-----------	--	--	--	--	-------	-------

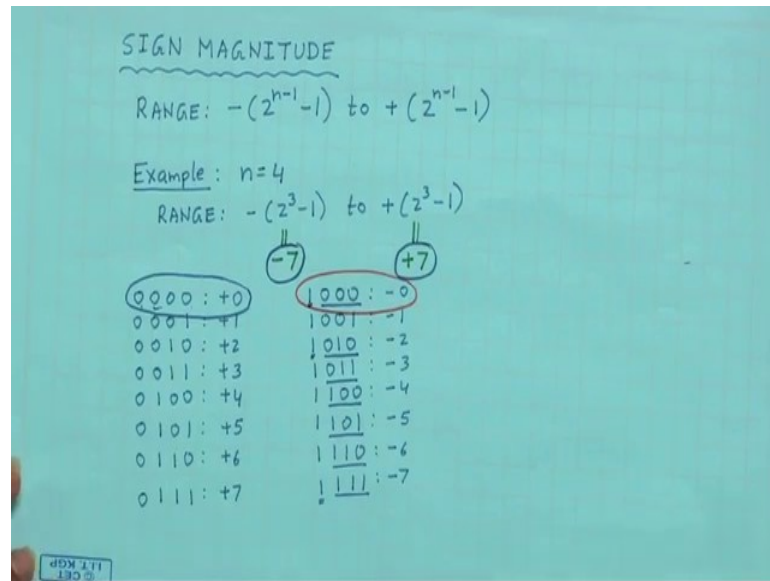
← Sign (under b_{n-1}) ← Magnitude (under b_{n-2} to b_0) →

- A problem: Two different representations for zero.
+0: 0 00..000 and -0: 1 00..000

Logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA are visible at the bottom.

First come to sign-magnitude representation. So, in a sign-magnitude representation for an n-bit number representation, the most significant bit indicates the sign. So, if the most significant bit is 0, then we say it is a positive number; if it is 1 it is a negative number, and the remaining n-1 bits represent the magnitude of that number. So, for an n-bit number, what will be the range that can be represented, it is $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$.

(Refer Slide Time: 28:39)




So, let us see this for a 4-bit number. So, for a 4-bit number, you can see the range is starting from -7 to +7. So, let us see how we can represent -7 to +7. So, 0 will be same as 0, and so on till 7. And starting from this is -0 because the first bit represent the sign and this is the magnitude, and then we have -1, -2, etc.

So, this is how we can represent the set of numbers for $n = 4$ using sign magnitude representation. So, these are all the magnitude and this is the sign of the number. Now, what is the problem here, the problem here is that you see you have two representations of 0. So, you have -0 and you have +0. So, this is one of the demerit of this method, we have two different representations for 0.

(Refer Slide Time: 30:31)

(b) Ones Complement Representation

- Basic idea:
 - Positive numbers are represented exactly as in sign-magnitude form.
 - Negative numbers are represented in 1's complement form.
- How to compute the 1's complement of a number?
 - Complement every bit of the number (1→0 and 0→1).
 - MSB will indicate the sign of the number (0: positive, 1: negative).



Let us move on with another representation that is 1's complement representation. So, the basic idea here is positive numbers are represented exactly in sign magnitude form; and for the negative numbers these are represented in 1's complement. In 1's complement, in each number if a bit is 1 it will be converted to 0; if it is 0 it will be converted to 1. And the MSB will indicate the sign of the number; for a positive number it will be 0, for a negative number it will be 1.

(Refer Slide Time: 31:09)


Example for n=4

Decimal	1's complement	Decimal	1's complement
+0	0000	-7	1000
+1	0001	-6	1001
+2	0010	-5	1010
+3	0011	-4	1011
+4	0100	-3	1100
+5	0101	-2	1101
+6	0110	-1	1110
+7	0111	-0	1111

To find the representation of, say, -4, first note that

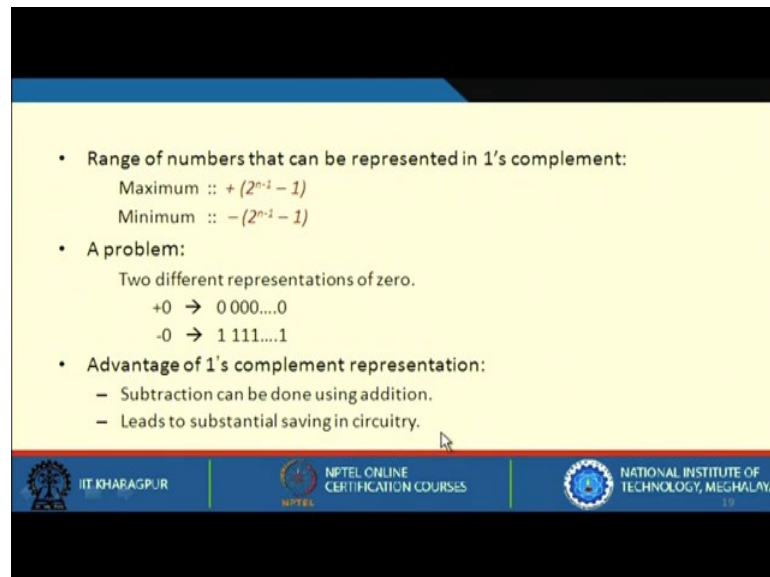
+4 = 0100

-4 = 1's complement of 0100 = 1011



So, this is the representation for $n = 4$. From 0 to 7, it will be the same as sign magnitude; and starting from -7 the representation is different. So, let us take the -4. How do you represent 4? We represent 4 as 0100. So, -4 is represented using 1s complement as 1011. But here also the same problem exists; we have two representations of 0.

(Refer Slide Time: 32:04)



The slide contains the following text:

- Range of numbers that can be represented in 1's complement:
Maximum :: $+(2^{n-1} - 1)$
Minimum :: $-(2^{n-1} - 1)$
- A problem:
Two different representations of zero.
 $+0 \rightarrow 0\ 000\dots 0$
 $-0 \rightarrow 1\ 111\dots 1$
- Advantage of 1's complement representation:
 - Subtraction can be done using addition.
 - Leads to substantial saving in circuitry.

The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA, along with the number 19.

Here also the range of numbers that can be represented is $+(2^{n-1}-1)$ to $-(2^{n-1}-1)$, but the problem here is we have two representations of +0 and -0. But there are few advantages; that is, using 1's complement subtraction can be done using addition and it leads to substantial saving in circuitry. So, you will be look into all these things in detail in the Arithmetic and Logic Unit.

(Refer Slide Time: 33:13)

(c) Twos Complement Representation

- Basic idea:
 - Positive numbers are represented exactly as in sign-magnitude form.
 - Negative numbers are represented in 2's complement form.
- How to compute the 2's complement of a number?
 - Complement every bit of the number ($1 \rightarrow 0$ and $0 \rightarrow 1$), and then *add one* to the resulting number.
 - MSB will indicate the sign of the number (0: positive, 1: negative).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us move on. And let us see 2's complement representation. Here the positive numbers are represented exactly the way as sign magnitude form, but the negative numbers are represented in 2's complement form. By 2's complement what we mean? Firstly, we do 1's complement and then we add 1 to the resulting number. So, this is the 2's complement representation. Here also the MSB will indicate the sign of a number; 0 for positive, 1 for negative.

(Refer Slide Time: 33:53)

Example for n=4

Decimal	2's complement	Decimal	2's complement
+0	0000	-8	1000
+1	0001	-7	1001
+2	0010	-6	1010
+3	0011	-5	1011
+4	0100	-4	1100
+5	0101	-3	1101
+6	0110	-2	1110
+7	0111	-1	1111

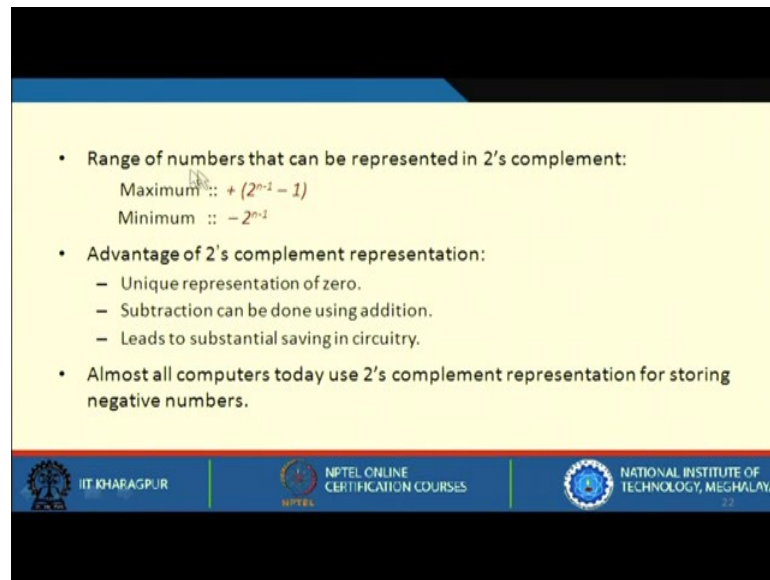
To find the representation of, say, -4, first note that

$$+4 = 0100$$
$$-4 = 2\text{'s complement of } 0100 = 1011 + 1 = 1100$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, this is the representation for $n = 4$. Now, we see that from 0 to 7 the representation is same. The negative number representations are also shown. And the main advantage is that here there is only single representation of 0. So, what is the range of number that we can have, one more than the previous one.

(Refer Slide Time: 35:01)



The slide contains the following text:

- Range of numbers that can be represented in 2's complement:
 - Maximum :: $+(2^{n-1} - 1)$
 - Minimum :: -2^{n-1}
- Advantage of 2's complement representation:
 - Unique representation of zero.
 - Subtraction can be done using addition.
 - Leads to substantial saving in circuitry.
- Almost all computers today use 2's complement representation for storing negative numbers.

The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

So, the range of numbers that can be represented in 2's complement is -2^{n-1} to $+(2^{n-1}-1)$. We have a unique representation of 0, subtraction can be done using addition, and it leads to substantial saving in circuitry. Almost all computers today use 2's complement representation for storing negative numbers.

(Refer Slide Time: 35:40)

• Some other features of 2's complement representation

a) Weighted number representation, with the MSB having weight -2^{n-1} .

-2^{n-1}	2^{n-2}	2^1	2^0
b_{n-1}	b_{n-2}	b_1	b_0

$$D = -b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_22^2 + b_12^1 + b_02^0$$

b) Shift left by k positions with zero padding multiplies the number by 2^k .

00010011 = +19	:: Shift left by 2 ::	01001100 = +76
11100011 = -29	:: Shift left by 2 ::	10001100 = -116

IIT KHARAGPUR
 NPTEL ONLINE CERTIFICATION COURSES
 NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, let us also see some more features of 2's complement, which will be necessary for the further lecture units. It represents a weighted number representation where MSB having the weight -2^{n-1} . So, what does it represent, it represent that all these numbers will have $2^0, 2^1, 2^{n-2}$, but the last one the magnitude will be -2^{n-1} .

(Refer Slide Time: 36:22)

-20 : 101100 in 2's complement

-2^5	2^4	2^3	2^2	2^1	2^0		
	0	1	1	0	0	= $-2^5 + 2^3 + 2^2 = -20$	
-2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	1	1	0	1	0	0	
						= $-2^6 + 2^5 + 2^3 + 2^2 = -20$	
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1	1	0	1	0	0	0
							= $-2^7 + 2^6 + 2^5 + 2^3 + 2^2 = -20$

⇒ SIGN EXTENSION IN 2'S COMPLEMENT

So, let us see this. So, how we can represent -20 in 2's complement? This is 2^0 , this is 2^1 , this is 2^2 , this is 2^3 , 2^4 , and -2^5 . So, for any negative number that we represent using 2's complement the most significant bit will have 1.

Now, let us add an extra 1 to it, and let us see that whether this changes or not. So, we added 1 more in the most significant bit and let us calculate it once more. The value remains -20. So, even if you add 1 more 1 into it, this will not change, this will be represented as $-2^7 + 2^6$, and this will remain as -20. So, this is what we call sign extension in 2's complement.

Now, what happens if we shift left by k position with 0 padding we are shifting left with 0 padding? What it does it multiplies the number by 2^k . So, we can take an example where this is the number what we say we shift left. So, we are shifting it left. So, this number is shifted to left by 2. So, we will shift it once and we shift it once more. If you shift it two times, it will become 01001100 and this is equivalent to +76 --- basically we have multiplied by 2^2 that is 4. Similarly, if this is -29, we shift left by 2 and we get -116. So, shifting left meaning multiplication by 2.

(Refer Slide Time: 39:24)

c) Shift right by k positions with sign bit padding divides the number by 2^k .

00010110 = +22 :: Shift right by 2 :: 00000101 = +5
 11100100 = -28 :: Shift right by 2 :: 11111001 = -7

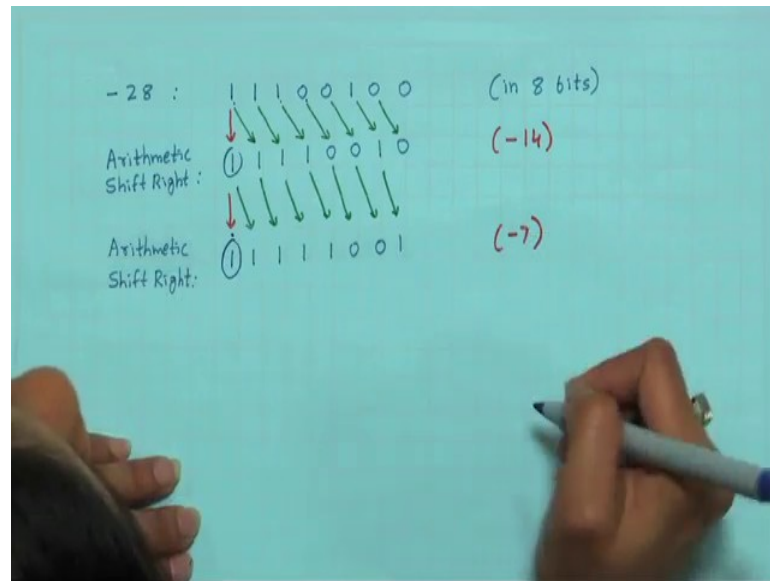
d) The sign bit can be copied as many times as required in the beginning to extend the size of the number (called *sign extension*).

X = 00101111 (8-bit number, value = +47)
 Sign extend to 32 bits:
 00000000 00000000 00000000 00101111

X = 10100011 (8-bit number, value = -93)
 Sign extend to 32 bits:
 11111111 11111111 11111111 10100011

Now, we will also see shift right by k positions with sign bit padding, what will happen divides the number by 2^k . Let us take an example. So, this example is where we shift right by 2 positions and then what is happening. So, here you have a number +22, once we divide by 2 we will get 11. Again we divide by 2 we will get 5. Similarly, -28.

(Refer Slide Time: 40:08)



So, let us take this example of -28 here. So, -28 can be represented as 11100100 in 8-bits. And then we do an arithmetic right shift. When we do an arithmetic right shift the most whatever value is a most significant bit that comes here and all other bits are shifted 1111 position see that is what we have done. So, all other bits are shifted to 1111 position and then this bit comes in the most significant bit and this representation is nothing but -14. Similarly, we are doing arithmetic shift right once more and same 1 bit comes in this position. So, 1 is coming here and all other bits are getting shift to 111 position to the right. So, we get 1111001, this is equivalent to -7. So, this is how we perform shift right by k position with sign bit and which actually divides the number by 2^k .

Now, this sign bit can be copied as many times as required I have already shown you this in the beginning to extend the size of the number called, this is called sign extension. So, this is the 8-bit number which is +47; sign extension to 32-bit you add as many number as 0 as required it will still remain +47. Similarly, a negative number -93, we can add as many 1 in the beginning to extend it to 32-bit the number will remain as -93 only. So, we can see that in 2's complement how we can represent it and how we can do sign extension.

So, we come to the end of lecture 6, where we briefly talked about number systems such that it will be easy for you to get into how instructions are represented and get executed in next lectures.

Thank you.